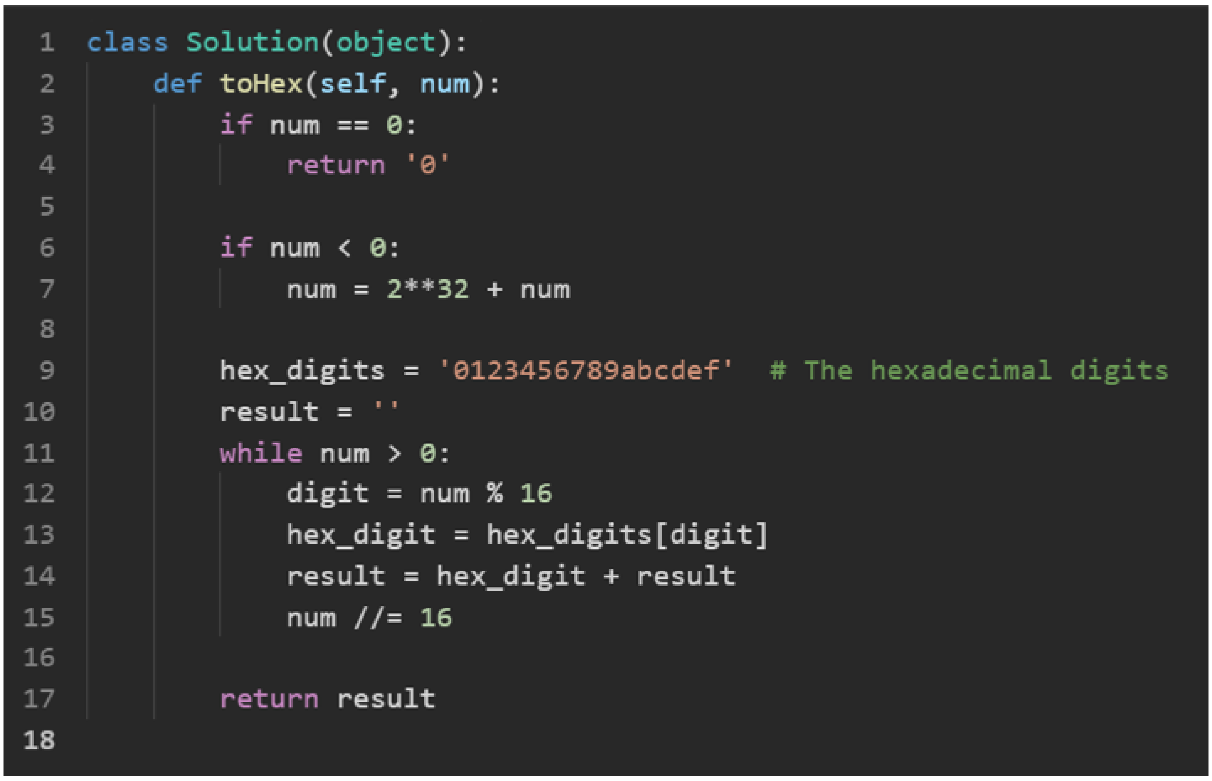
Coding Challenges

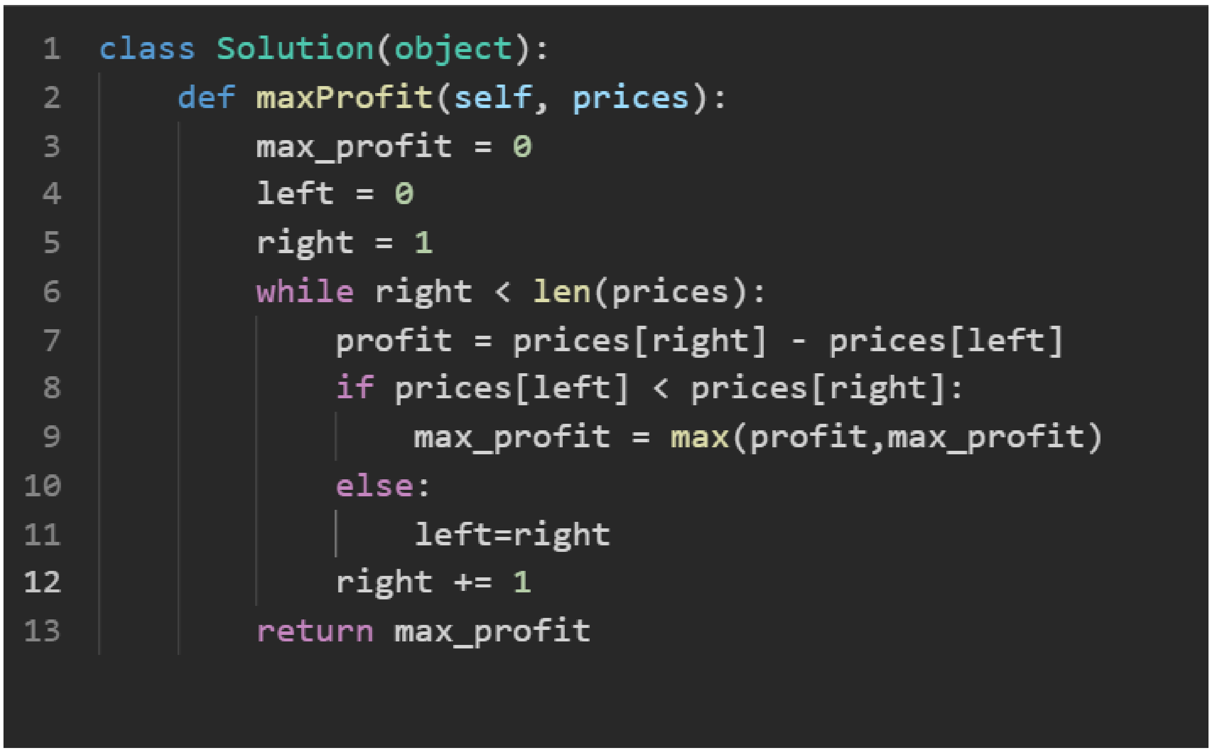
1- Convert a Number to Hexadecimal



```python
class Solution(object):
    def toHex(self, num):
        if num == 0:
            return '0'

        if num < 0:
            num = 2**32 + num

        hex_digits = '0123456789abcdef'  # The hexadecimal digits
        result = ''
        while num > 0:
            digit = num % 16
            hex_digit = hex_digits[digit]
            result = hex_digit + result
            num //= 16

        return result

```

First we need to check if the number is 0 so return 0 , then check if it is negative if true we apply 2’s complement by adding 2 power 32 (no of bits of int number) , create a string which have all possible values of hex number and create string to hold the result , finally apply decimal to hex algorithm => Divide the number by 16, Get the integer quotient for the next iteration, Get the remainder for the hex digit, Repeat the steps until the quotient is equal to 0. We use the remainder to get its equivalent hex number from hex_digits string , then concatenate result with new hex digit but it must be in right side to append new digit from the left.

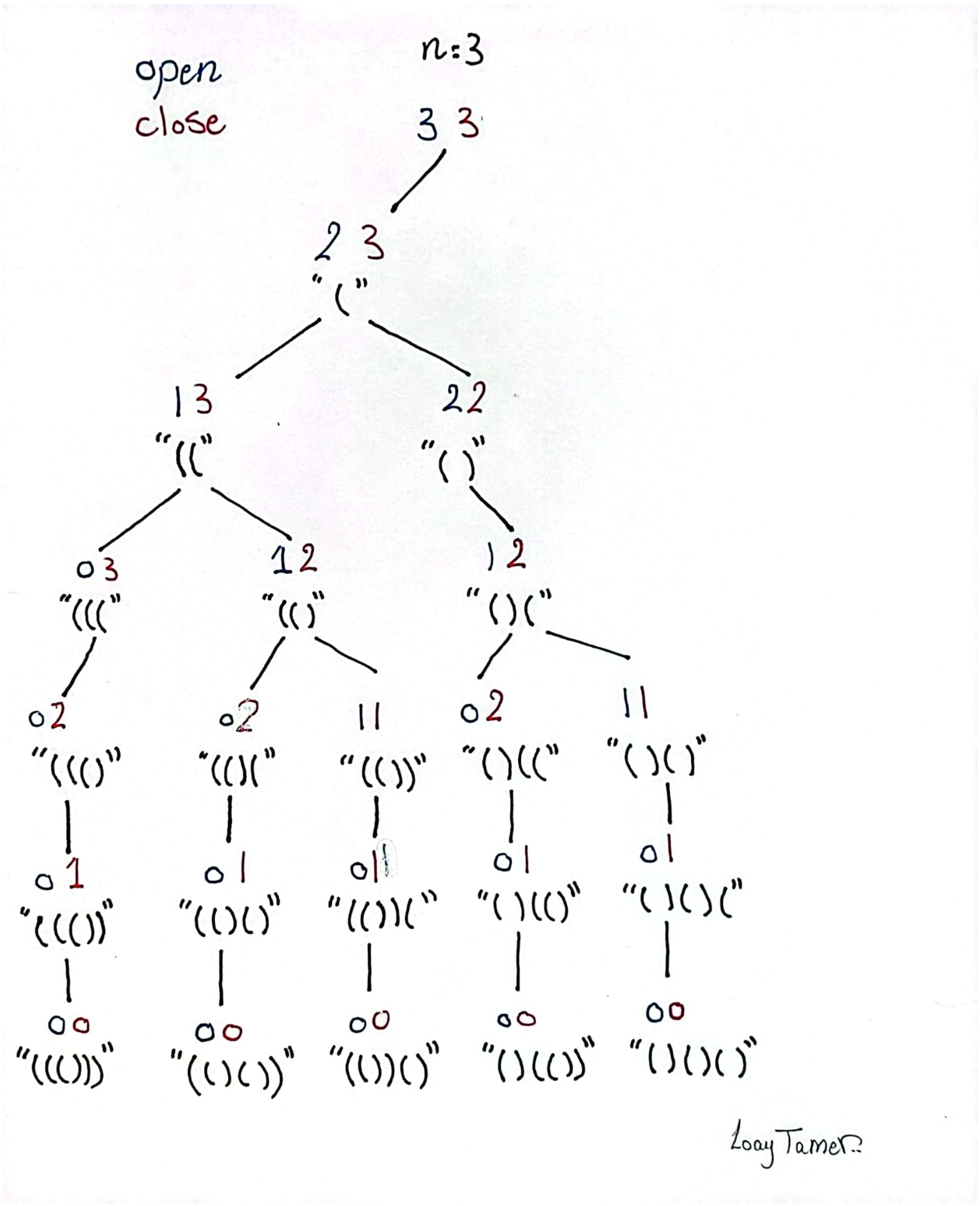
2 - Best Time to Buy and Sell Stock:



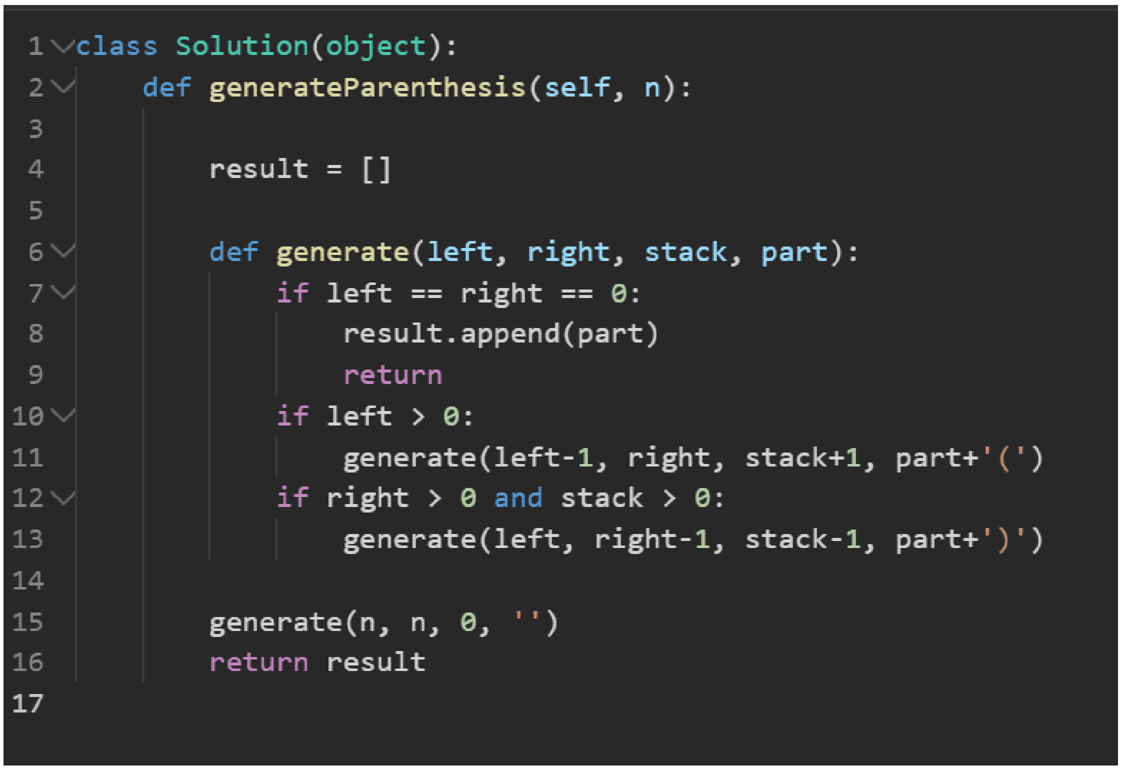
```python
class Solution(object):
    def maxProfit(self, prices):
        max_profit = 0
        left = 0
        right = 1
        while right < len(prices):
            profit = prices[right] - prices[left]
            if prices[left] < prices[right]:
                max_profit = max(profit,max_profit)
            else:
                left=right
            right += 1
        return max_profit
```

To find max profit we can use two pointers algorithm , first pointer (left) points to buying day, second pointer (right) points to selling day , looping on items calculate the profit and check if buying less than selling take the max between stored max_profit and new profit, if not increment position of left and in any case we increment the right position to complete our check .

3 - Generate Parentheses



If n = 3 that means that we have 3 opening parentheses and 3 closing parentheses , when left and right (open and close) are equal we must open a bracket,now we have 2 options : first open a new bracket, second close the bracket. To control going to depth we will use a stack variable, when we open a bracket it will increment its value to let us know if we can write a closing bracket or not.



```python
class Solution(object):
    def generateParenthesis(self, n):

        result = []

        def generate(left, right, stack, part):
            if left == right == 0:
                result.append(part)
                return
            if left > 0:
                generate(left-1, right, stack+1, part+'(')
            if right > 0 and stack > 0:
                generate(left, right-1, stack-1, part+')')

        generate(n, n, 0, '')
        return result

```

First we create an empty list to store our answers , then define a generate function that generates our strings that takes left, right, stack, part(item to append to our answer).
Our base case (that will break the recursion function) when left and right equal to zero, our recursion checks if the left greater than 0 it will concatenate ‘(‘ with part string and recall the function again with decrement left and increment stack value, the remaining part of recursion checks that right greater than zero and stack greater than zero (to ensure that we write an open brackets and close them) if true we recall the function again with decrement of right and stack value and concatenate ‘)’ with part string.finally , calling the function and return the result.